REMARKS

It is postulated in the final rejection, despite the fact that the drawings in the cited Vila reference are inconsistent with the premise of the rejection, that the Vila reference teaches other embodiments different from those depicted.

The first cited support for this premise is column 7, lines 18-40. However, that material expressly references Figures 9, 11, and 13. Thus, rather than supporting a different mode of operation than what is depicted in the figures, it further illustrates and substantiates it. Further, in the paragraph beginning at line 36 of column 7, the operation shown in the figures is discussed. Specifically, it can be seen in reference to Figure 11 that initially no data begins to be stored at any time in any of the lanes until a test sequence is received. In other words, the write pointer does not increment until a test sequence is received. The first data after the test sequence is always put in the leftmost position. As shown in Figures 11-13, the read pointer never moves.

Nothing in any of the material cited in column 7 suggests that the read pointer ever moves. The read pointer stays in the leftmost position and the write pointer moves one position after the test sequence is received for each new data. Thus, the discussion in column 7, lines 20-25, about the pointers advancing is simply the operation described in the figures wherein the read pointer never moves, but the write pointer for each lane advances one place for new data that is received after the test sequence. At line 27 of column 7, where it is indicated that while Figures 9 and 11-13 are depicted as initially containing no data, the buffers need not be cleared upon the occurrence of a reset event. Because the read and write pointers are configured to allow only storage locations containing valid data to be read, it does not matter what values are stored in the buffer in normal operation, they will be overwritten with valid data before they are read out of the buffer. All this means is that the buffers do not need to be empty. Whether they contain data or not, they will be overwritten starting from the left after receiving the test sequence.

The paragraph beginning at line 36 talks about a reset event. There, the pointers associated with each buffer are reset. Each of the pointers are set to indicate a predetermined initial storage location. The read and write pointers are maintained in those initial positions until the test sequence is received. This is exactly what is shown in the figures. Rather than suggesting some mode of operation inconsistent with the figures, this material simply explains the figures. In every case, the read pointer is always at the leftmost position. The write pointer is the only one that

moves. Thus, the claimed limitation which requires that data be delivered from different positions within two serial buffers relative to one another cannot be met by the Vila reference.

Secondly, it is suggested in the office action on page 6 that Vila teaches an embodiment where the read pointer may be inhibited or other storage elements may be used to perform the de-skewing process, thus, allowing the read pointer to be placed in the second position. Cited in support of this second premise is column 10, lines 60-65. It is suggested in line 60 that, as another example, the buffers may selectively inhibit reads and writes of data to the buffers rather than manipulate pointers. All this says is that, rather than use a pointer to decide where the read is to occur and where the write is to occur, inhibiting may be utilized to inhibit locations that are not to be read or not to be written. But this in no way suggests that the operation would be in any other way different. In other words, reading would always occur from the leftmost position and writing would be incremented location-by-location each time new data is received after the test sequence.

Further, it is explained in yet another example, FIFOs rather than circular buffers may be utilized in which the read and write pointers are manipulated to control. Of course, just because you have a non-circular buffer, does not change much of anything in the cited reference. Figures 11-13 show linear representations of the buffers. Thus, whether or not they are circular, those buffers still always read from the same position and increment the write pointers as new data comes in.

Thus, there is no support for the proposition that the reference teaches anything but always reading from exactly the same corresponding location in each buffer. Therefore, the reference fails to meet the claims and reconsideration is mandated.

Respectfully submitted,

Date: October 6, 2005

Timothy N. Trop, Reg. No. 28,994
TROP, PRUNER & HU, P.C.
8554 Katy Freeway, Ste. 100
Houston, TX 77024
713/468-8880 [Phone]
713/468-8883 [Fax]

Attorneys for Intel Corporation